CLAIMS:

What is claimed is:

1. A method in a data processing system that implements an object-oriented software environment for translating method calls to version-specific method calls, said method comprising the steps of:

providing an interface to an underlying object, applications utilizing said interface to communicate with said underlying object, said interface being separate from said underlying object;

generating a plurality of version-specific underlying objects, each one of said version-specific underlying objects being a different version of said underlying object;

generating a translation object for communicating between said interface and each one of said version-specific underlying objects;

utilizing said translation object for translating an interface method call invoked on said interface to a version-specific method call for said underlying object for each version of said underlying object; and

generating said translation object from a single proxy class and a single invocation handler class, wherein the same proxy class and invocation handler class are used to generate said translation object for each different version of said underlying object.

2. The method according to claim 1, further comprising the steps of:

determining a current version of said underlying object;

determining whether said translation object exists for said current version; and

in response to determining that said translation object does not exist, generating said translation object utilizing a building object.

3. The method according to claim 2, further comprising the steps of:

creating said building object utilizing said proxy class and said invocation handler class.

4. The method according to claim 1, further comprising the steps of:

generating a building instance of said proxy class and said invocation handler class;

utilizing said building instance to generate an invocation handler instance for said interface;

utilizing said building instance to generate said translation object; and

including said invocation handler instance in said translation object.

5. The method according to claim 1, further comprising the steps of:

generating a translation instance for a particular version of said underlying object;

including an instance of said proxy class and an instance of said invocation handler class;

invoking said interface method call on said translation instance;

passing said interface method call from said proxy instance to said invocation handler instance; and

invoking a version-specific method call on said particular version of said underlying object using said invocation instance, said version-specific method call being equivalent to said invoked interface method call.

6. The method according to claim 5, further comprising the steps of:

utilizing, by said invocation handler instance, said interface method call and original parameters included in said interface method call to locate said version-specific method call;

passing said original parameters to said invoked version-specific method call;

receiving, by said invocation handler instance, a return object from version-specific method call; and

returning, by said invocation handler instance, said return object.

7. The method according to claim 6, further comprising the steps of:

invoking said interface method call on said translation instance by a client object; and

returning, by said invocation handler instance, said return object to said client object.

8. The method according to claim 5, further comprising the steps of:

utilizing, by said invocation handler instance, said interface method call and original parameters included in said interface method call to locate said version-specific method call;

determining whether any of said original parameters are proxy objects; and

in response to determining that at least one of said original parameters are proxy objects, replacing each said proxy object with an underlying object type for said particular version.

9. The method according to claim 7, further comprising the steps of:

determining whether said return object is a second interface; and

in response to a determination that said return object is said second interface, creating a second proxy instance that implements a return type for said second interface.

10. The method according to claim 7, further comprising the steps of:

determining whether said return object is a second interface;

in response to determining that said return object is said second interface, determining whether said return type is an array; and

in response to determining that said return object is an array, creating a separate proxy instance that implements a return type for said second interface for each element of said array.

11. The method according to claim 1, further comprising the steps of:

said translation object being a single object for inserting translation code for translating said interface method call and for manipulating return parameters received after execution of said translated interface method call.

12. A data processing system that implements an object-oriented software environment for translating method calls to version-specific method calls, comprising:

an interface to an underlying object, applications utilizing said interface to communicate with said underlying object, said interface being separate from said underlying object;

a plurality of version-specific underlying objects, each one of said version-specific underlying objects being a different version of said underlying object;

a translation object for communicating between said interface and each one of said version-specific underlying objects;

said translation object being utilized for translating an interface method call invoked on said interface to a version-specific method call for said underlying object for each version of said underlying object; and

said translation object being generated from a single proxy class and a single invocation handler class, wherein the same proxy class and invocation handler class are used to generate said translation object for each different version of said underlying object.

13. The system according to claim 12, further comprising:

said data processing system including a CPU executing code for determining a current version of said underlying object;

said CPU executing code for determining whether said translation object exists for said current version; and

in response to determining that said translation object does not exist, said translation object being generated utilizing a building object.

14. The system according to claim 13, further comprising:

said building object being created utilizing said proxy class and said invocation handler class.

15. The system according to claim 12, further comprising:

a building instance of said proxy class and said invocation handler class;

said building instance being utilized to generate an invocation handler instance for said interface;

said building instance being utilized to generate said translation object; and

including said invocation handler instance in said translation object.

16. The system according to claim 12, further comprising:

a translation instance being generated for a particular version of said underlying object;

including an instance of said proxy class and an instance of said invocation handler class;

said interface method call being invoked on said translation instance;

said interface method call being passed from said proxy instance to said invocation handler instance; and

a version-specific method call being invoked on said particular version of said underlying object using said invocation instance, said version-specific method call being equivalent to said invoked interface method call.

17. The system according to claim 16, further comprising the steps of:

said invocation handler instance utilizing said interface method call and original parameters included in said interface method call to locate said version-specific method call;

said original parameters being passed to said invoked version-specific method call;

said invocation handler instance receiving a return object from version-specific method call; and

said invocation handler instance returning said return object.

18. The system according to claim 17, further comprising:

said interface method call being invoked on said translation instance by a client object; and

said invocation handler instance returning said return object to said client object.

19. The system according to claim 16, further comprising:

said invocation handler instance utilizing said interface method call and original parameters included in said interface method call to locate said version-specific method call;

said CPU executing code for determining whether any of said original parameters are proxy objects; and

in response to determining that at least one of said original parameters are proxy objects, said CPU executing code for replacing each said proxy object with an underlying object type for said particular version.

20. The system according to claim 18, further comprising:

said CPU executing code for determining whether said return object is a second interface; and

in response to a determination that said return object is said second interface, said CPU executing code for creating a second proxy instance that implements a return type for said second interface.

21. The system according to claim 18, further comprising:

said CPU executing code for determining whether said return object is a second interface;

in response to determining that said return object is said second interface, said CPU executing code for determining whether said return type is an array; and

in response to determining that said return object is an array, said CPU executing code for creating a separate proxy instance that implements a return type for said second interface for each element of said array.

22. The system according to claim 12, further comprising:

said translation object being a single object for inserting translation code for translating said interface method call and for manipulating return parameters received after execution of said translated interface method call.

23. A computer program product in a data processing system that implements an object-oriented software environment for translating method calls to version-specific method calls, said product comprising:

instruction means for providing an interface to an underlying object, applications utilizing said interface to communicate with said underlying object, said interface being separate from said underlying object;

instruction means for generating a plurality of version-specific underlying objects, each one of said version-specific underlying objects being a different version of said underlying object;

instruction means for generating a translation object for communicating between said interface and each one of said version-specific underlying objects;

instruction means for utilizing said translation object for translating an interface method call invoked on said interface to a version-specific method call for said underlying object for each version of said underlying object; and

instruction means for generating said translation object from a single proxy class and a single invocation handler class, wherein the same proxy class and invocation handler class are used to generate said translation object for each different version of said underlying object.

24. The product according to claim 23, further comprising:

instruction means for determining a current version of said underlying object;

instruction means for determining whether said translation object exists for said current version; and

in response to determining that said translation object does not exist, instruction means for generating said translation object utilizing a building object.

25. The product according to claim 24, further comprising:

instruction means for creating said building object utilizing said proxy class and said invocation handler class.

26. The product according to claim 23, further comprising:

instruction means for generating a building instance of said proxy class and said invocation handler class;

instruction means for utilizing said building instance to generate an invocation handler instance for said interface;

instruction means for utilizing said building instance to generate said translation object; and

instruction means for including said invocation handler instance in said translation object.

27. The product according to claim 23, further comprising:

instruction means for generating a translation instance for a particular version of said underlying object;

instruction means for including an instance of said proxy class and an instance of said invocation handler class;

instruction means for invoking said interface method call on said translation instance;

instruction means for passing said interface method call from said proxy instance to said invocation handler instance; and

instruction means for invoking a version-specific method call on said particular version of said underlying object using said invocation instance, said version-specific method call being equivalent to said invoked interface method call.

28. The product according to claim 27, further comprising:

instruction means for utilizing, by said invocation handler instance, said interface method call and original parameters included in said interface method call to locate said version-specific method call;

instruction means for passing said original parameters to said invoked version-specific method call;

instruction means for receiving, by said invocation handler instance, a return object from version-specific method call; and

instruction means for returning, by said invocation handler instance, said return object.

29. The product according to claim 28, further comprising:

instruction means for invoking said interface method call on said translation instance by a client object; and

instruction means for returning, by said invocation handler instance, said return object to said client object.

30. The product according to claim 27, further comprising:

instruction means for utilizing, by said invocation handler instance, said interface method call and original parameters included in said interface method call to locate said version-specific method call;

instruction means for determining whether any of said original parameters are proxy objects; and

in response to determining that at least one of said original parameters are proxy objects, instruction means for replacing each said proxy object with an underlying object type for said particular version.

31. The product according to claim 29, further comprising:

instruction means for determining whether said return object is a second interface; and

in response to a determination that said return object is said second interface, instruction means for creating a second proxy instance that implements a return type for said second interface.

32. The product according to claim 29, further comprising:

instruction means for determining whether said return object is a second interface;

in response to determining that said return object is said second interface, instruction means for determining whether said return type is an array; and

in response to determining that said return object is an array, instruction means for creating a separate proxy instance that implements a return type for said second interface for each element of said array.

33. The product according to claim 23, further comprising:

said translation object being a single object for inserting translation code for translating said interface method call and for manipulating return parameters received after execution of said translated interface method call.